



Regroupement en familles de produits en fonction des outils disponibles

Jean-Marie Proth

► To cite this version:

Jean-Marie Proth. Regroupement en familles de produits en fonction des outils disponibles. [Rapport de recherche] RR-0370, INRIA. 1985, pp.20. inria-00076186

HAL Id: inria-00076186

<https://hal.inria.fr/inria-00076186>

Submitted on 24 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



CENTRE DE ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
BP 105
78153 Le Chesnay Cedex
France
Tél (3) 954 90 20

Rapports de Recherche

N° 370

REGROUPEMENT EN FAMILLES DE PRODUITS EN FONCTION DES OUTILS DISPONIBLES

Jean Marie PROTH

Mars 1985

REGROUPEMENTS EN FAMILLES DE PRODUITS
EN FONCTION DES OUTILS DISPONIBLES

Jean Marie PROTH

PROJET SAGEP

* Ce travail a été effectué dans le cadre d'un contrat passé avec l'Agence de l'Informatique.



PAPIER RECUPERÉ ET RECYCLÉ

ABSTRACT

This paper gives an algorithm which leads to a partition of a set of n objects in p classes ($p \ll n$). The objective is to minimize the total of the inertias of the p classes. We show a way to obtain a good initial situation.

RESUME

Ce papier donne un algorithme qui conduit à une partition d'un ensemble de n objets en p classes ($p \ll n$). L'objectif est de minimiser la somme des inerties des p classes. Nous montrons une méthode pour obtenir une bonne situation initiale.

I. INTRODUCTION

Les lignes qui suivent ont pour but de proposer un algorithme de classement de n objets en p classes ($p \leq n$).

Après avoir posé le problème nous montrons que, partant de p points quelconques, il est possible de construire une suite de partitions de l'ensemble des objets en p classes et que cette suite converge. Simultanément, nous montrons que la somme des moments d'inertie des classes constituant la partition diminue lorsque le rang de l'itéré augmente.

Nous utiliserons ensuite ce résultat pour bâtir un algorithme de classification automatique qui débute par un processus de génération automatique des p points initiaux.

Nous proposons ensuite les programmes (écrits en FORTRAN) et un exemple d'application.

II. ENONCE DU PROBLEME

Nous considérons n éléments de \mathbb{R}^m et une distance sur \mathbb{R}^m . Nous notons x_i , $i=1, \dots, n$, ces éléments et d la distance. u_i désignera le poids associé à x_i .

Nous souhaitons trouver une partition de $E = \{x_1, x_2, \dots, x_n\}$ en p sous-ensembles, chacun de ces sous-ensembles regroupant des points proches au sens de la distance d . Nous avons choisi d'essayer de minimiser la somme des inerties de ces sous-ensembles, c'est à dire :

$$\sum_{k=1}^p \sum_{i/x_i \in e^k} u_i d^2(x_i, y^k) \quad (1)$$

où :

y^k est le centre d'inertie de e^k , $k=1, 2, \dots, p$
et $\{e^1, e^2, \dots, e^p\}$ est la partition cherchée.

Nous verrons dans la suite qu'il est possible de trouver un minimum local de (1) et que le choix de la partition initiale laisse espérer que ce minimum local est voisin du minimum global.

III. RESULTAT PRINCIPAL

Nous proposons d'abord un algorithme qui, partant d'une situation initiale quelconque, aboutit à un minimum local. Cet algorithme est donné dans le théorème suivant.

THEOREME

Pour $i=1, 2, \dots, n$, soit x_i un point de \mathbb{R}^m et u_i le poids associé à ce point. Soit encore d une distance sur \mathbb{R}^m .

Nous désignerons par E l'ensemble $\{x_1, x_2, \dots, x_n\}$.

Soient $x_1^0, x_2^0, \dots, x_p^0$ ($p \leq n$) des points de \mathbb{R}^m appartenant ou non à E .

On considère l'algorithme suivant, que nous désignons par A1 :

1. Faire $k=0$.
2. Construire une partition $e_1^k, e_2^k, \dots, e_p^k$ de E de la manière suivante :

pour $i=1, 2, \dots, n$:

a. Rechercher l'ensemble $J_i \subset \{1, 2, \dots, p\}$ défini comme suit :

$$J_i = \{j \in \{1, 2, \dots, p\} / d(x_i, x_j^k) = \min_{s=1, \dots, p} d(x_i, x_s^k)\}$$

b. Si $\text{card}(J_i)=1$, x_i est affecté à e_j^k , sinon x_i est affecté à $e_{j_1}^k$ où :

$$j_1 = \min_{j \in J_i} j$$
3. Pour $j=1, 2, \dots, p$, on cherche x_j^{k+1} , centre d'inertie de e_j^k . On rappelle que ce point vérifie :

$$\sum_{i/x_i \in e_j^k} u_i d^2(x_j^{k+1}, x_i) = \min_{y \in \mathbb{R}^m} \left[\sum_{i/x_i \in e_j^k} u_i d^2(y, x_i) \right] \quad (3)$$
4. Test.

4.1. Si $x_j^{k+1} = x_j^k$ pour $j=1, 2, \dots, p$, fin de processus.

4.2. Sinon :

4.2.1. Faire $k=k+1$.

4.2.2. Aller en 2.

L'algorithme A1 converge.

Démonstration.

a. Nous notons :

$I(y, E) = \sum_{x \in E} u_x d^2(y, x)$, où u_x est le poids associé à x et E un ensemble fini.

Si $x_j^{k+1} \neq x_j^k$ pour au moins un $j \in \{1, 2, \dots, p\}$, alors

$$\sum_{j=1}^p I(x_j^{k+1}, e_j^k) < \sum_{j=1}^p I(x_j^k, e_j^k) \quad (\text{voir (3)}) \quad (4)$$

et :

$$\sum_{j=1}^p I(x_j^k, e_j^k) \leq \sum_{j=1}^p I(x_j^k, e_j^{k-1}) \quad (5)$$

(voir (2) et la définition du moment d'inertie)

Finalement :

$$\sum_{j=1}^p I(x_j^{k+1}, e_j^k) < \sum_{j=1}^p I(x_j^k, e_j^{k-1}) \quad (6)$$

Cette inégalité montre que si, pour deux pas d'itération successifs, l'ensemble des centres d'inertie est modifié, alors la somme des inerties des éléments de la partition décroît, donc la partition est modifiée et la nouvelle partition est différente de toutes les partitions précédemment rencontrées.

b. E étant fini, l'ensemble de ses partitions est fini. Associée à la conclusion de a., cette remarque achève la démonstration. \square

Remarques :

1. Le théorème que nous venons de présenter indique la convergence de l'algorithme proposé, mais n'affirme pas que la partition obtenue est celle qui assure le minimum absolu de la somme des inerties. Le contre exemple suivant nous en convaincra. Les points x_1 , x_2 et x_3 sont de poids 1. Les distances sont indiquées sur la figure 1. Nous considérons la partition constituée des sous-ensembles $\{x_1, x_3\}$ et $\{x_2\}$ ($p=2$).

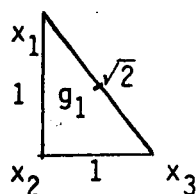


Fig. 1

Les centres d'inertie de ces sous-ensembles sont respectivement g_1 et x_2 et la somme des inerties est $1/2$. Il est facile de vérifier que cette partition ne sera pas modifiée par l'algorithme et, en même temps, que la partition $\{x_1\}, \{x_2, x_3\}$ conduirait à une somme d'inerties égale à $1/4$.

2. L'approche précédente est du type "nuées dynamiques" (voir en particulier [2]).
3. L'algorithme ci-dessus suppose fixé le nombre de classes. C'est une hypothèse forte.
4. Les résultats obtenus sont très dépendants des points initiaux choisis dans \mathbb{R}^m et de la distance.

Nous proposons dans le paragraphe suivant un algorithme que conduit à une situation initiale raisonnable.

IV. RECHERCHE D'UNE SITUATION RAISONNABLE

Choisissons un réel positif r et appelons densité d'un point x_i ($i \in \{1, 2, \dots, n\}$) de \mathbb{R}^m le nombre de points dont la distance à x_i est inférieure ou égale à r .

Nous pensons qu'un choix raisonnable des p points initiaux doit porter sur des points :

- 1) de forte densité, ce qui assure qu'un grand nombre de points sont proches des points choisis.
- 2) convenablement répartis dans l'ensemble des points à classer.

A noter que ces deux critères peuvent être incompatibles.

Nous proposons l'algorithme suivant, noté A2, pour le choix des points initiaux :

1. Choix de r .
2. Choix de $\delta \geq 2r$.
3. Choix (éventuellement au hasard) de $x_1^0 = \{x_1, x_2, \dots, x_n\} \in E$.
4. Simultanément, par balayage des points de E :
 - 4.1. Recherche de la densité de x_1^0 .
 - 4.2. Recherche de E_1 , ensemble des points de E distants de x_1^0 de plus de δ .
5. Itérations :
 - 5.1. Faire $k=1$.
 - 5.2. Choix (au hasard) de $x_{k+1}^0 \in E_k$.
 - 5.3. Simultanément, par balayage des points de E_k :
 - 5.3.1. Recherche de la densité de x_{k+1}^0 .
 - 5.3.2. Recherche de E_{k+1} , ensemble des points de E_k distants de x_{k+1}^0 de plus de δ .
 - 5.4. Test.
 - 5.4.1. Si $E_{k+1} = \emptyset$.
 - 5.4.1.1. Si $k+1 \geq p$, aller en 6.
 - 5.4.1.2. Sinon :
 - 5.4.1.2.1. Edition d'un message : "Nombre de points initiaux insuffisants".
 - 5.4.1.2.2. Décision de l'utilisateur :
 - . soit retour en 1.
 - . soit retour en 2.
 - . soit fin de processus.
 - 5.4.2. Si $E_{k+1} \neq \emptyset$.
 - 5.4.2.1. Faire $k=k+1$.
 - 5.4.2.2. Retour en 5.2.
6. Classement des x_i^0 , $i=1,2,\dots,k+1$, par ordre décroissant de leur densité.
7. Les p premiers points obtenus sont les points initiaux.

Remarques :

1. A chaque pas d'itération, nous choisisons au hasard un point suffisamment éloigné de tous les points précédemment retenus et nous calculons sa densité. Ce faisant, nous limitons le nombre de densités à calculer.
2. L'importance de la densité apparaît au cours de la seconde phase du processus (classement), qui conduit à ne retenir que les points de plus forte densité parmi les points éloignés les uns des autres de plus de δ .
3. Le nombre de points retenus doit être supérieur ou égal au nombre de points initiaux. Si ce nombre est proche du nombre de points initiaux, on privilégie la répartition des points initiaux dans l'ensemble des points à classer au détriment des densités. S'il est très grand, on privilégie les densités.

V. LE LOGICIEL DE CLASSIFICATION

Le logiciel de classification que nous présentons maintenant est structuré de manière à pouvoir être complété pour autoriser le choix de plusieurs distances et de plusieurs processus de classification à partir de ces distances.

Les données à traiter sont dans le fichier 70 qui est structuré comme suit :

Premier enrégistrement :

Nombre d'observations	I4
Nombre de paramètres	I4

Enrégistrement suivants :

Chaque enrégistrement contient une suite de 50 observations (sauf peut être le dernier).

Chaque observation est composée :

-de la suite des paramètres	F2.0
-du poids de l'observation	F2.0

Le programme est écrit pour recevoir sans modification des observations composées de 19 paramètres au maximum, plus le poids, l'ensemble étant exprimé en format F2.0.

Une modification du format entraînerait une modification du recl (record length) du fichier 70.

Une modification du nombre maximal de paramètres modifierait en conséquence le "recl" du fichier 70 ainsi que :

les dimensions des tableaux actuellement égales à 20 dans le programme principal et ses sous-programmes dist et rep (cette dimension doit toujours être supérieure de 1 au nombre de paramètres).

Une modification du nombre maximal d'observations modifierait en conséquence le "recl" du fichier 70 ainsi que :

1. la dimension du tableau iz dans le programme principal et ses sous-programmes.
2. plus indirectement, la dimension du tableau it qui doit être supérieure ou égale au nombre de points candidats au rôle de points initiaux pour la constitution des classes.

La dimension 50 correspond au nombre d'observations par enrégistrement logique.

Nous donnons dans les pages suivantes le programme principal et les sous-programmes.

Les résultats des calculs figurent dans le fichier 50.

A l'entrée du programme, le nombre d'observations et de paramètres sont affichés.

L'utilisateur fournit :

1. le nombre de classes souhaitées.
2. la distance utilisée. Pour l'instant, la distance euclidienne et la distance égale au nombre de non correspondances (cas d'observations linéaires) sont seules disponibles.

3. les représentants des classes. Actuellement, le choix se réduit aux centres d'inertie.
4. le rayon servant au calcul des densités. C'est le rayon des boules ayant comme centre les points-observation et qui servent à la détermination de la densité des points-observation : la densité d'un point-observation est égale au nombre de points contenus dans la boule dont il est le centre.
5. la distance minimale entre deux points initiaux. Cette distance est déterminante. Trop importante, elle interdit la détermination d'un nombre suffisant de points initiaux. Trop faible, elle peut conduire à choisir des points initiaux trop proches.

Le programme fournit, dans le fichier 50 et pour chaque classe :

1. le représentant de la classe.
2. la liste des éléments de la classe.

Nous donnons le programme principal, les deux sous-programmes auxquels il fait appel :

1. le sous-programme "dist" qui calcule la distance entre deux observations.
2. le sous-programme "rep" qui calcule les représentants des classes.

PROGRAMME PRINCIPAL

```

6 c      le fichier 70 est deja rempli pour l'exemple demande
7 c      le fichier 50 -> resultats
8 c
9 c  CONTRAINTES ACTUELLES DU PROGRAMME
10 c---Nombre maxi. d'observations : 1999
11 c---Nombre maxi. de classes      : 50
12 c---Nombre maxi. de parametres   : 19
13      dimension a(50,20),aini(50,20),iz(2000),ds(50),it(500),b1(20),b2(20)
14      open(70,access="direct",form="formatted",recl=2000)
15      open(50,form="formatted")
16      read(70,1,rec=1,err=27)n,m
17 c
18      eps=.001
19      write(0,70)n,m
20      70 format("_Nombre d'observations",i4," _Nombre de parametres",i4)
21 c
22      1 format(2i4)
23      21 write(0,5)
24      5 format(2x,"_Nombre de classes ? ")
25      read(0,3)ip
26      3 format(v)
27      write(0,200)
28      200 format(12x,"<< CHOIX DES DISTANCES POUR LA CONSTITUTION DES CLASSES
29      &>>",//)
30      write(0,201)
31      201 format(2x,"_Distance euclidienne ;.....taper 1 ",//)
32      write(0,202)
33      202 format(2x,"_Dissemblance separant les 0 ;taper 2 ")
34      read(0,3)idis
35      write(0,400)
36      400 format(12x,"<< CHOIX DES REPRESENTANTS >>")
37      write(0,401)
38      401 format(2x,"centre d'inertie; tapez 1")
39      read(0,3)irep
40 c
41 c
42 c      *****
43 c      *          PHASE D'INITIALISATION          *
44 c      *****
45 c
46 c      Recherche du premier point.
47 c
48      write(0,2)
49      2 format(2x,"_Choix du rayon pour le calcul des densites. ")
50      read(0,3)r
51      8 write(0,4)
52      4 format(2x,"_Choix de la distance minimale entre points initiaux. ")

```

```

53      read(0,3)dl
54      if(dl.lt.r) goto 8
55      call random_uniform(x)
56      if(x.gt.(1.-1.e-4))x=1.-1.e-4
57      k=x*n+1
58      k1=(k-.5)/50+2
59      k2=k-(k1-2)*50
60      k3=n-(k1-2)*50
61      if(k3.gt.50) k3=50
62      read(70,6,rec=k1,err=13)((a(k,j),j=1,m),k=1,k3)
63      6 format(1000f2.0)
64      jj=1
65      do 7 j=1,m
66      b1(j)=a(k2,j)
67      7 aini(1,j)=a(k2,j)
68 c
69 c                      Recherche de la densite du premier
70 c                      point et des points suivants.
71 c
72      do 12 j=1,50
73      12 ds(j)=0
74      ii=0
75      do 9 k=1,n
76      k1=(k-.5)/50+2
77      k2=k-(k1-2)*50
78      if(k2.gt.1)go to 10
79      k3=n-(k1-2)*50
80      if(k3.gt.50) k3=50
81      read(70,6,rec=k1,err=26)((a(i,j),j=1,m),i=1,k3)
82      10 do 11 j=1,m
83      b2(j)=a(k2,j)
84      11 continue
85      call dist(b1,b2,m,idis,y)
86      if(y.le.r) ds(jj)=ds(jj)+1
87      if(y.lt.dl) goto 9
88      ii=ii+1
89      it(ii)=k
90      9 continue
91 c
92 c
93 c                      *****
94 c                      *              ITERATIONS              *
95 c                      *****
96 c
97 c                      Choix du point suivant
98 c
99      18 call random_uniform(x)
100      if(x.gt.(1.-1.e-4))x=1.-1.e-4
101      ii1=x*ii+1
102      k=it(ii1)
103      k1=(k-.5)/50+2
104      k2=k-(k1-2)*50
105      k3=n-(k1-2)*50
106      if(k3.gt.50) k3=50
107      read(70,6,rec=k1,err=28)((a(k,j),j=1,m),k=1,k3)
108      jj=jj+1
109      do 14 j=1,m
110      b1(j)=a(k2,j)
111      aini(jj,j)=a(k2,j)
112      14 continue

```

```

113 c
114 c
115 c               Recherche de la densite du point
116 c               et des points a retenir.
117 c
118         ii1=0
119         do 15 k5=1,ii
120           k=it(k5)
121           k1=(k-.5)/50+2
122           k2=k-(k1-2)*50
123           k3=n-(k1-2)*50
124           if(k3.gt.50) k3=50
125           read(70,6,rec=k1,err=29)((a(i,j),j=1,m),i=1,k3)
126           do 17 j=1,m
127     17 b2(j)=a(k2,j)
128           call dist(b1,b2,m,idis,y)
129           if(y.le.r) ds(jj)=ds(jj)+1
130           if(y.lt.dl) goto 15
131           ii1=ii1+1
132           it(ii1)=k
133     15 continue
134           ii=ii1
135           if(ii.gt.0) goto 18
136           if(jj.ge.ip) goto 22
137           write(0,19)
138     19 format(2x,"_Nombre de points initiaux insuffisant .")
139           write(0,20)
140     20 format(2x,"_Abandon = 0"/,2x,"_Reprise = 1"/)
141           read(0,3)is
142           if(is.eq.0) goto 1664
143           goto 21
144 c
145 c
146 c               *****
147 c               *   CLASSEMENT DES POINTS INITIAUX   *
148 c               *                   POSSIBLES                   *
149 c               *****
150 c
151 c
152     22 do 23 i=1,jj-1
153         do 23 j=i+1,jj
154           if(ds(i).gt.ds(j))goto 23
155           do 24 k=1,m
156             x=aini(i,k)
157             aini(i,k)=aini(j,k)
158     24 aini(j,k)=x
159           x=ds(i)
160           ds(i)=ds(j)
161           ds(j)=x
162     23 continue
163 c
164 c
165 c               *****
166 c               *   FIN DE L'INITIALISATION   *
167 c               *****
168 c
169 c               Edition des points initiaux.
170 c
171         do 25 i=1,ip
172         write(50,5000)i

```



```

173 5000 format(10x,"<< POINT ",i4," >>")
174      write(50,5001)(aini(i,k),k=1,m)
175 5001 format(5(2x,e14.7))
176      write(50,5050)
177 5050 format(10x,"Densite")
178      write(50,5002)ds(i)
179 5002 format(15x,e14.7)
180      25 continue
181      index=0
182 c
183 c
184 c      *****
185 c      *      CONSTITUTION DES CLASSES      *
186 c      *****
187 c
188 c
189      333 do 40 i=1,n
190          40 iz(i)=0
191              do 41 k=1,n
192                  k1=(k-.5)/50+2
193                  k2=k-(k1-2)*50
194                  if(k2.gt.1) goto 42
195                  k3=n-(k1-2)*50
196                  if(k3.gt.50) k3=50
197                  read(70,6,rec=k1,err=39)((a(i5,j),j=1,m),i5=1,k3)
198          42 do 43 j=1,m
199              43 b1(j)=a(k2,j)
200                  do 44 i=1,ip
201                      do 45 j=1,m
202                          45 b2(j)=aini(i,j)
203                              call dist(b1,b2,m,idis,y)
204                              if(i.gt.1) goto 46
205                              z=y
206                              iz(k)=i
207                              goto 44
208          46 if(y.gt.z) goto 44
209              z=y
210              iz(k)=i
211          44 continue
212          41 continue
213          write(0,485)
214 485      format(3x,"PRODUIT",2x,"CLASSE")
215          do 486 i=1,n
216 486      write(0,487)i,iz(i)
217 487      format(5x,i4,6x,i4)
218 c
219 c
220 c      *****
221 c      *      RECHERCHE DES REPRESENTANTS      *
222 c      *****
223 c
224 c      Recherche des representants suivants.
225 c
226      call rep(ip,iz,aini,irep,n,m,idis,itel,eps)
227      index=index+1
228      if(index.lt.30) goto 600
229      write(0,31)
230      31 format(2x,"_Nombre d'iterations superieur a 30 .")
231      goto 700
232 600 if(itel.eq.1) goto 333

```

```

233 c
234 c
235 c
236 c
237 c
238 c
239 700 do 701 i=1,ip
240     write(50,801)i
241 801 format(15x,"_Classe numero ",i4)
242     write(50,899)
243 899 format(5x,"Representant")
244     write(50,898)(aini(i,j),j=1,m-1)
245 898 format(5(1x,e14.7))
246     do 702 k=1,n
247         k1=(k-.5)/50+2
248         k2=k-(k1-2)*50
249         if(k2.gt.1) goto 704
250         k3=n-(k1-2)*50
251         if(k3.gt.50) k3=50
252         read(70,6,rec=k1,err=30)((a(i5,j),j=1,m),i5=1,k3)
253 704 if(iz(k).ne.i) goto 702
254         write(50,802)k,(a(k2,j),j=1,m)
255 802 format(2x,i4/2x,20(f2.0,1x))
256 702 continue
257 701 continue
258     go to 1664
259 30 write(0,32)
260 32 format("erreur label 30")
261     goto 1664
262 29 write(0,33)
263 33 format("erreur label 29")
264     goto 1664
265 28 write(0,34)
266 34 format("erreur label 28")
267     goto 1664
268 26 write(0,35)
269 35 format("erreur label 26")
270     goto 1664
271 27 write(0,36)
272 36 format("erreur label 27")
273     goto 1664
274 13 write(0,37)
275 37 format("erreur label 13")
276     go to 1664
277 39 write(0,38)
278 38 format("erreur label 39")
279 1664 close(70)
280     close(50)
281     stop
282     end

```

SOUS-PROGRAMME "dist"

```
283 c
284 c
285 c
286      subroutine dist(b1,b2,m,idis,y)
287      dimension b1(20),b2(20)
288      y=0
289      goto (1,2),idis
290      1 do 10 j=1,m-1
291      10 y=y+(b1(j)-b2(j))**2
292      y=y**0.5
293      goto 100
294      2 do 20 j=1,m-1
295      i5=1
296      i5=b1(j)+b2(j)
297      if(i5.eq.2) i5=0
298      20 y=y+i5
299      100 return
300      end
```

SOUS-PROGRAMME "rep"

```

418 c
419 c
420 c
421     subroutine rep(ip,iz,aini,irep,n,m,idis,itel,eps)
422     dimension a(50,20),aini(50,20),iz(2000),b1(20),b2(20),co(50,20,3)
423     goto (1,100),irep
424     1 do 2 i=1,50
425       do 2,j=1,m-1
426         do 2 k=1,3
427           2 co(i,j,k)=0
428             do 12 i=1,50
429               do 12 j=1,20
430 12      aini(i,j)=0.
431             do 3 i=1,n
432               k1=(i-.5)/50+2
433               k2=i-(k1-2)*50
434               if(k2.gt.1) goto 4
435               k3=n-(k1-2)*50
436               if(k3.gt.50) k3=50
437 c
438       read(70,5,rec=k1,err=9)((a(k,j),j=1,m),k=1,k3)
439       5 format(1000f2.0)
440       4 i5=iz(i)
441       do 6 j=1,m-1
442         co(i5,j,1)=co(i5,j,1)+a(k2,m)
443         co(i5,j,2)=co(i5,j,2)-2*a(k2,m)*a(k2,j)
444       6 co(i5,j,3)=co(i5,j,3)+a(k2,m)*a(k2,j)**2
445       3 continue
446       itel=0
447       do 7 i=1,ip
448         if(co(i,1,1).gt.eps)go to 17
449         write(0,18)i
450 18      format(2x,"Representant num.",i4,"non utilise")
451         go to 7
452 17      do 7 j=1,m-1
453         a1=co(i,j,1)
454         a2=co(i,j,2)
455         a3=co(i,j,3)
456         pp=-a2/2./a1
457         uu=abs(pp-aini(i,j))
458         if(uu.gt.eps) itel=1
459         aini(i,j)=pp
460       7 continue
461       goto 1000
462 100 write(0,15)
463     15 format(2x,"_Coefficient de dissimilarite non prevu !")
464     go to 1000
465     9 write(*,8)
466     8 format(2x,"_Erreur lecture fichier 70 dans sous-programme rep.")
467 1000 return
468 end

```

Dans sa version actuelle, ce logiciel est prévu pour traiter des paramètres qui prennent leurs valeurs dans des intervalles proches. On ne normalise donc pas les valeurs prises par les paramètres.

VI. UN EXEMPLE D'APPLICATION

Nous considérons un atelier composé de m machines et susceptible de traiter n produits différents. Nous sommes dans le cas d'un job-shop, c'est à dire dans le cas où chaque type de produit passe sur une partie seulement des machines de l'atelier, et dans des ordres variables suivant le type de produit.

Nous cherchons à regrouper les produits qui utilisent des moyens "proches". Un tel regroupement sera appelé "famille".

Les données du problème sont donc regroupées dans un tableau à n lignes et m colonnes :

$$A = (a_{ij}), \quad i=1,2,\dots,n \text{ et } j=1,2,\dots,m$$

avec :

$$a_{ij} = \begin{cases} 1 & \text{si le produit } i \text{ passe sur la machine } j. \\ 0 & \text{sinon.} \end{cases}$$

La figure 2 donne le tableau de l'exemple que nous avons traité. Il comporte 20 produits et 12 paramètres.

Nombre d'individus 20 Nombre de parametres 12
 Le poids est en derniere position

1	1.1.1.1.0.0.0.0.0.0.0.0.2.
2	0.0.0.0.1.1.1.0.0.0.0.0.3.
3	0.0.0.0.0.0.0.0.1.1.0.0.0.4.
4	0.0.0.0.0.0.0.0.0.0.1.1.1.4.
5	0.0.0.0.0.0.0.0.1.1.0.0.0.3.
6	0.0.0.0.1.1.1.0.0.0.0.0.0.2.
7	1.1.1.1.0.0.0.0.0.0.0.0.0.3.
8	0.0.0.0.0.0.0.0.0.0.0.1.1.1.3.
9	1.1.1.1.0.0.0.0.0.0.0.0.0.2.
10	0.0.0.0.0.0.0.0.0.1.1.0.0.0.3.
11	0.0.0.0.0.0.0.0.0.0.0.1.1.1.2.
12	1.1.1.1.0.0.0.0.0.0.0.0.0.4.
13	0.0.0.0.1.1.1.0.0.0.0.0.0.5.
14	0.0.0.0.0.0.0.0.0.1.1.0.0.0.2.
15	0.0.0.0.0.0.0.0.0.0.0.1.1.1.2.
16	0.0.0.0.0.0.0.0.0.1.1.0.0.0.3.
17	1.1.1.1.0.0.0.0.0.0.0.0.0.4.
18	0.0.0.0.1.1.1.0.0.0.0.0.0.4.
19	0.0.0.0.0.0.0.0.0.0.0.1.1.1.3.
20	1.1.1.1.0.0.0.0.0.0.0.0.0.2.

Fig. 2

A l'exécution, nous avons fourni un rayon de 1 et une distance minimale entre points initiaux de 2. Nous avons demandé 3 classes.

Compte tenu de la simplicité des données, le résultat a été obtenu en une itération.

Le système a retenu les points initiaux :

1 1 1 1 0 0 0 0 0 0 0 0 pour la classe 1.

Ce point a une densité de 6.

0 0 0 0 0 0 0 1 1 0 0 0 pour la classe 2.

Ce point a une densité de 5.

0 0 0 0 0 0 0 0 1 1 1 pour la classe 3.

Ce point a une densité de 5.

Finalement, la classification automatique nous a conduit à (en utilisant la distance euclidienne) :

Classe 1

Représentant : 1 1 1 1 0 0 0 0 0 0 0 0

Numéros des produits de la classe : 1, 7, 9, 12, 17 et 20.

Classe 2

Représentant : 0 0 0 0 0,48 0,48 0,48 0,52 0,52 0 0 0

Numéros des produits de la classe : 2, 3, 5, 8, 10, 13, 14, 16, 18.

Classe 3

Représentant : 0 0 0 0 0 0 0 0 0 1 1 1

Numéros des produits de la classe : 4, 8, 11, 15, 19.

CONCLUSION

Ce travail est un premier pas dans la voie de la technologie de groupe. Le programme que nous présentons a été écrit pour obtenir rapidement des familles de produits en fonction des outils utilisés. Le poids associé à chaque produit représente le nombre moyen de produits de ce type à fabriquer durant une période de temps donnée. Bien entendu, on peut remplacer les machines par des opérations. Les produits sont alors regroupés en fonction de la similitude de transformations qu'ils subissent.

Notons encore que la nécessité de fixer à priori le nombre de classes est une contrainte importante. A notre connaissance, personne n'a encore réussi à se dégager de cette contrainte même si, dans certains cas, le problème est déplacé.

La démarche suivante, que nous présenterons dans un prochain rapport, aura pour but de regrouper les machines (ou opérations) dont l'activité essentielle est consacrée à une même famille de produits afin d'obtenir des îlots de fabrication.

BIBLIOGRAPHIE

- [1] J.MINOT - Y.LEMOINE - B.MUTEL,
"Implantation assistée par ordinateur de la Technologie de Groupe",
Congrès AFCET Productique et Robotique Avancée -Besançon -Novem-
bre 83.
- [2] Ensemble des travaux de E.DIDAY et collaborateurs.

